Performance analysis of server selection schemes for Video on Demand servers

THESIS SUBMITTED IN PARTIAL FULFILLMENT FOR

THE DEGREE OF

Bachelor of Technology

in

Computer Science and Engineering

By

Suraj Dash

Roll no:107CS019

Alok Kumar Prusty

Roll no:107CS063

Under the Guidance of

Prof. Bibhudatta Sahoo

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela-769 008, Orissa, India

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis titled **"Performance analysis of server selection schemes for Video on Demand servers "** submitted by **Suraj Dash : 107CS019** and **Alok Kumar Prusty : 107CS063** in the partial fulfillment of the requirement for the degree of Bachelor of Technology in Computer Science Engineering, National Institute of Technology, Rourkela , is being carried out under my supervision.

To the best of my knowledge the matter embodied in the thesis has not been submitted to any other university/institute for the award of any degree or diploma.

Prof. Bibhudatta Sahoo

date : Department of Computer Science and Engineering

National Institute of Technology

ACKNOWLEDGEMENT

We wish to express our sincere and heartfelt gratitude towards our guide **Prof. Bibhudatta Sahoo**, Computer Science Engineering Department, for his guidance, sympathy, inspiration and above all help in all regards during the duration of my project.

We would also like to thank all the professors of the department of Computer Science and Engineering, National Institute of Technology, Rourkela, for their constant motivation and guidance.

Submitted by:

Suraj Dash

Roll no-107CS019

Alok Kumar Prusty

Roll no-107CS063

Computer Science and Engineering

National Institute of Technology

Rourkela

Contents

List of Figures

Abstract

Web Services have gained considerable attention over the last few years. This is due to increase in use of the Internet which results in increased web traffic. Web servers find applications in E-commerce and Video-on-Demand(VOD) systems which have resulted in speedy growth of the web traffic. Therefore the concept of load balancer aimed to distribute the tasks to different Web Servers to reduce response times was introduced. Each request was assigned a Web Server decided by the load balancer in such a way that tasks were uniformly distributed among the available servers. Server selection algorithms are aimed to meet the QoS for interactive VoD.This thesis attempts to analyze the performance of FCFS, Randomized, Genetic algorithms and Heuristics algorithms for selecting server to meet the VoD requirement . Performance of these algorithms have been simulated with parameters like makespan and average resource utilization for different server models. This thesis presents an efficient heuristic called Ga-max-min for distributing the load among different servers. Heuristics like min-min and max-min are also applied to heterogeneous server farms and the result is compared with the proposed heuristic for VOD Servers. Ga-max-min was found to provide lower makespan and higher resource utilization than the genetic algorithm.Extensive simulations have been carried out by the simulator designed using MATLAB R2010a.

Chapter 1

Introduction

1 Introduction

1.1 Introduction

Web server is a program that provides content like web pages over the world wide web. The simultaneous open connections to the web server are generally limited. Thus the waiting time becomes high when the number of requests to the web server is large resulting in DOS (Denial of Service) attack. An effective solution to this problem is the use of multiple servers known as clustered Web Servers or a server farm. Multimedia communications require continuous service, i.e. read, process and transfer the information should be done with minimum delay which is vastly improved if we use a server farm.

The performance of a server farm depends on the type of routing, server capacity and scheduling policies used. The server capacity can be homogeneous or heterogeneous. In case of homogeneous systems, each of the servers in the server farm are of equal capacity and the request is processed by the server having the least number of tasks in the queue, i.e. Join the shortest queue policy[3].Heterogeneous systems scores over homogeneous systems if tasks are of different sizes. Heterogeneous systems can also include task-specific systems, i.e. for more computation oriented tasks we can use an array processor.

Load Balancing Policy consists of load index policy, information collection policy, task location and task transfer policy. In our approach we assume that the nature of task coming to the web server is known beforehand. Load index policy keeps track of the number of tasks in the queue and information collection policy has the knowledge about the type of tasks coming to the server farm and the nature of web traffic distribution. This can be done by checking the server log file and obtain information like average page views, busy times, visit duration and the most requested page by the customer. Task transfer policy decides whether the task has to be serviced in the local servers or sent to other servers located remotely. Our main focus is on the task location policy which describes scheduling algorithm for the various tasks. We also assume an infinite capacity front end dispatcher which assigns the tasks to various servers.

In this paper we examine the different scheduling algorithms, First come first serve, ran-

dom and genetic algorithm. The metric for comparing different algorithms is makespan. Makespan is defined as the maximum time taken to complete all the tasks given to the dispatcher or load balancer. An advantage for using genetic approach is that there is no need to set any threshold values on the number of tasks or utilization of the server. The server load can be represented by the following equation[21]

Bandwidth = Average Daily Visitors x Average Page Views x Average Page Size x 31 x Fudge Factor (1)

If people are allowed to download files from the site, the bandwidth calculation becomes: **Bandwidth= [(Average Daily Visitors x Average Page Views x Average Page Size) + (Average Daily File Downloads x Average File Size)] x 31 x Fudge Factor** (2)

- **Average Daily Visitors** - The number of people expected to visit a site, on average, each day. It may vary significantly on the basis of how a site is marketed.
- **Average Page Views** It represents the average number of web pages visited by a person.
- **Average Page Size** It shows the average size of the web pages, expressed in kilobytes(KB)
- **Average Daily File Downloads** - The number of downloads expected to occur from a site.It depends on number of visitors and average downloads per visitor.
- **Average File Size** - Average size of files that are downloadable from the site.
- **Fudge Factor** - A number greater than 1. A fudge factor of 1.5 implies that the estimate is off by 50Usually, bandwidth is offered in terms of Gigabytes (GB) per month. Hence the entire formula is multiplied by 31.

We then focus on a particular application of web servers: Video on Demand. VOD servers are different from normal web servers because they demand a consistent and higher data rate. They find applications in Video Conference (VC), IP telephony, Multimedia Mail, Multimedia Mall, Digital Libraries [9]. The demand for on demand video services have increased significantly in the recent years and is expected to rise further due to advancement in technology to meet the high Qos required by VOD applications. In fact, commercial VoD services with complete video cassette recorder (VCR) functions have appeared. However, owing to ever increasing user demands, when the user access rates increase, several issues need to be tackled, e.g., high block rate, long startup delay, service interruption, frame losing[.The Qos as desired by the users are generally subjective in nature. So they must be mapped to an appropriate objective (quantitative) parameter so that we get a technically correct application.

1.2 Distributed multimedia requirements

This section briefly discusses the platforms and technologies requirements for distributed multimedia applications and finally highlights on the specific requirements of VoD service .The requirements [9,20] are summarized as follows:

(i) *Application Programming Interfaces:* We need portable user interfaces, smart agents and conference management.

(ii) *Audio Quality:* Using conventional speaker phones (usually half duplex) with fullmotion video, the result is disturbing; the participants can see another persons lips moving but cannot hear them .A full-duplex echo canceling speaker phone is a good choice.

(iii) *Video Quality:* Quality is often inadequate, and it is much less than the broadcast quality. Parallax-free viewing (i.e. direct eye contact) and proper face lighting, along with NTSC image quality, are preferred. Full-size faces as opposed to talking heads are preferred. Higher quality cameras could be used as a tradeoff to compensate for blinding the user with extra light.

(iv)*Multimedia Object Technology:* A database should now include audio, video and other media objects. Standard software with object data bases cannot meet large-scale VoD

requirements due to performance, real time constraints and object output controls.

(v) *Multimedia bridging:* Although some manufacturers have Multipoint Control Unit (MCU) for conference bridging on the market, the costs are still prohibitive for a small time provider of services.

(vi) *Standards:* Many of the industry providers are overzealous when they set high levels of expectation for quick deployment and successful penetration of applications. Simply investing in digitizing video and audio information is not the solution when there are multiple standards and proprietary methods that exist and compete with each other. Open standards give buyers a choice of vendors, offers a promise of compatibility, and gives an assurance that equipment will not quickly become obsolete. For manufacturers, this customer confidence means a larger market, leading to larger volumes, lower prices, and a greater variety of available products.

(vii) *Asynchronous Transfer Mode (ATM):* ATM technology provides fast packet transport and switching and multiplexing of packets. It supports synchronous and isochronous media, large bandwidth, flexible dynamic bandwidth, and supports standards. It supports speeds from OC-3(155.50 Mbps) to OC-12(622.08Mbps) speeds.

(viii) *Hardware:* There is need for a single VLSI chip for compression, decompression, CPU, Asynchronous Transfer Mode (ATM) segmentation and reassemble. Storage technology needs revitalization in improving the speeds and be able to have mass storage. Development of Super dense optical storage from Sony and Philips, Toshiba and Time Warner is useful.

(ix) *Cable Modems:* More than 30 million US households have PCs at home. Broadband CATV networks pass more than 90transport multimedia traffic by upgrading to hybrid fiber coax architectures. The key benefit of cable modems is that they are ten to hundred times faster than dial-up modems or ISDN. CATV is universally available and inexpensive. Companies such as Lucent Technologies, GI, ADC, NewBridge, Zenith and LANcity are developing cable modems.

(x) *Loop and access sub-network:* The ADSL technology is promising as it can deliver up to 6 Mbps using the currently available copper. The other technologies could be Switched Digital Video (SDV) and Multipoint Microwave Distribution System (MMDS).

Multimedia systems will impact computer systems, storage memory in terms of capacity, access time and transfer rate, interconnections (for example, bus bandwidth), processing power (as software codecs have to be processed), operating system, and network bandwidth. VOD could take up 173Mbps (i.e. 30frames/second x (480 x 525) pixels/frame x 24 bits/pixel) while video conferencing can take up to 69.6 Mbps (i.e. 30 frames per sec x (288 x 352) pixels/frame x 24 bits/pixel). Technological advancement in digital electronics and fiber optic communications are making more functions economical and simpler to use. The other contributing factor is the application software that is easy, and it should compel the user to continue using it. Creating multimedia applications that are easy and interesting requires inspiration and perspiration. Creating such an application will result in millions of users contributing to the final product. Now talking on the requirements of VOD service, it is characterized by asymmetric information flow as real-time signaling information transmitted to the head-end is less than what is transmitted to the set top box. A VOD database of compressed video requires standards such as MPEG-2. At a video rate of 3 Mbps, an average two-hour movie can occupy 2.7 GB in disk storage. A video server with 500 on-line titles would therefore need 1.35 terabytes of storage. Many video server manufacturers have chosen ATM to transport audio and video at 2.5 GBps. A significant requirement on video server is that its output data. rate should be 400 MB/s, and its storage will be 1.5TB of on-line disk capacity with 6TB of off-line archival tape. Some of the broad band transport requirements on One-way end-to-end delay, End-to-End delay jitter, differential delay, response time, Intra-media synchronization and Inter-media synchronization need to be satisfied.

1.3 Literature Review

Server Selection, Load balancing and scheduling issues have been studied quite extensively in the past. Most notable of the server selection algorithms [16] are the closest server algorithm that selects server based on the proximity to the client, optimized closest server algorithm that chooses the closest server among the free channels, Register all algorithm where the clients request is added to the queue of all the servers and Maximum-

MFQ-rank-first algorithm which computes the rank at the various server queues and assigns the request to the server having the best rank. There is also a minimum expected cost algorithm which based on parameters like latency and bandwidth computes the server with the minimum cost and a merging aware minimum expected cost algorithm which allocates server on the ability of the server to merge.

In light of the load balancing problems, Haight(1958), Halfin(1985), and Kingman(1961) are among the many people that studied join the shortest queue policy using two parallel servers with infinite buffer size. Gupta et al.[3] analyzed the join the shortest queue policy on processor sharing server farms. They used a single queue approximation and investigated the sensitivity of the queuing model to variations. Niyato et al. [4] studied load balancing for Internet video and audio server. They studied and compared various algorithms like Adaptive bidding, Diffusion and State change broadcast along with traditional round-robin and random algorithms. Wang et al. studied load balancing in heterogeneous systems, first considering two servers with different service rates and then extending their observations to multiple servers. This involved multiple thresholds setting which was done by heuristic methods. Ciardo et al. [6] devised a strategy for task allocation in web servers based on size distributions of the requested documents. Zhang et al.[2]analyzed the central load balancing model ,derived average response time and the rejection rate and compared three different routing policies.

The retrieval schemes for VOD can be classified into two categories, a) Disk level retrieval schemes[9.] which focuses on synchronizing and efficiently using the data between different storage devices and b)server level retrieval schemes[9.] which deliver data to the client whenever the need arise. Our approach is based on the server level. Similar requests can be batched or the server can be replicated [16] to achieve low latency and thus serve a higher number of requests. Server replication comes with an additional cost of installing new servers. Beyond a certain number of servers, further increase will only lead to more installation cost without improving the Qos like throughput, speedup etc. As shown in Figure 1 if the number of servers increases above 95-105 range then the makespan doesnt decrease further.

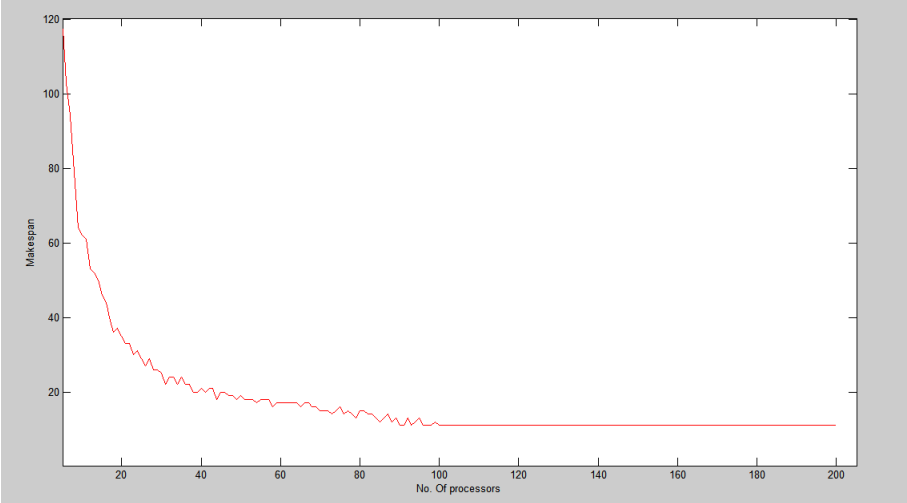

Figure 1: Depicting the makespan vs number of processors/servers.

1.4 Motivation

VOD networks followed centralized architecture in the early days. But with increase in number of requests the trend has shifted to distributed architecture for VOD networks. As the number of requests increases the number of servers required to cater to those request increases which adds to additional cost. If by some heuristics or means, we can efficiently allocate the tasks to the different available servers such that it optimizes the value of a metric like makespan and throughput, then the customer requirements can be met in a better manner.

VOD is a relatively new concept. Many of the existing load balancing algorithms hasnt been applied to VOD systems. Further, the need of proper server selection is necessary for maintaining high data rate (eg. 1.5Mbps for MPEG video) and minimizing the cost of service. The objective of this paper is twofold. Firstly to analyze the existing algorithms and heuristics in the context of VOD based systems, and secondly to analyze the performance of the proposed heuristic for two metrics namely makespan and Average resource utilization.

1.5 Problem statement

Let there be a task set T consisting of n(T1 T2... Tn) tasks and let there be m servers. The basic problem is where to map a task Ti among the m possible servers. This is done by the server selection strategy. The tasks should be allocated in such a way that after allocation of all the n tasks among the m servers, the performance metric should be optimized.

Many such algorithms have been proposed in the past. Generally the cost is measured as the major metric for server selection. But there are other factors like server load, response time that also impact the quality of service in a big way. There are various metrics available for comparing the server selection algorithms. Depending on the task set a particular metric or metrics are of more significance than others. It is not always advisable to allocate tasks according to a particular selection strategy if that strategy gives optimized solution for a single metric but can give undesirable results when evaluated against some other metric. So we need to decide which scheme gives the best performance for which metric and in doing that we are able to infer a scheme or a framework of schemes that is suitable for a particular nature of task.

1.6 Approaches to the problem

There are various approaches to solve the problem of server selection. Basically the algorithms on the basis of load balancing of server can be divided into two classes-traditional and heuristic methods. Some of the well known traditional algorithms are First Come First Serve(FCFS), Random and Genetic algorithms. Among the heuristic methods, the most commonly used are min-min and max-min.

The heuristics algorithms are not suitable for large data sets while the traditional Genetic algorithms are known to perform well for large data sets. So an approach should be used that involves dynamic selection of an algorithm based on the nature of the tasks at hand. This can be collected in a framework. The main switch to which all the request comes allocates the best possible algorithm according to the nature of the task set.

1.7 Thesis layout

The rest of the thesis is organized as follows. Section 2.2 of chapter 2 describes about the system model for a web server in general and VOD servers in specific and the sequence of activities for a VOD connection and retrieval. Section 2.3 suggests the various metrics used for comparison of performance of different algorithms. Section 2.4 compares various existing algorithms like First come first serve, Random and Genetic algorithms on the basis of Makespan. It also compares other heuristics like Min-min, Max-min and weighted mean time scheduling. Section 3.2 gives the description of genetic algorithm used and section 3.3 describes the max min algorithm. The proposed Ga-max-min algorithm is described in section 3.5. which uses Makespan and Average Resource utilization as a metric for comparison. Finally Chapter 4 concludes the result obtained in chapter 3 and future work is listed in chapter 5.

Chapter 2

Video-On-Demand Model And Performance Metric

2 Video-On-Demand: Model And Performance Metric

2.1 Introduction

Video on Demand servers are gaining popularity in the last few years and as a result of which it has been increasingly studied in the last few years. There are innumerable server selection strategies to optimize performance parameters of a VOD server. All these are designed keeping in view the architecture where it is implemented. A particular strategy with same input may give different output if applied to a different architecture. So the choice of network becomes extremely important.

Traditionally two broad types of network exist. Centralized and Distributed. A centralized architecture consists of a common server which serves all the requests. But as the number of request increases the load on central server increases and it becomes a single point for failure. On the other hand the load is distributed across many servers, on the basis of certain parameters like nature of movie, frequency of access, popularity etc. The distributed system is useful as compared to centralized system in three major aspects[9].

1) Good scalability At any point, the system can cater to a higher number of requests by increasing the number of servers.

2) High availabilityThe system can provide service even if some of the servers fail or are under maintenance.

3) Competitive performance-to-price ratio The distributed server configuration is adopted by a commercial VOD system called iTV system.

The distributed architecture is suitable for our problem statement as a centralized architecture doesnt demand the need for load balancing but is unsuitable for high number of tasks. The detailed server architecture is presented in the following sections.

2.2 System Architecture

The adapted Figure 2[17] depicts the prevalent 3-tier architecture for web servers. The main components are a set of web servers, a set of database server nodes and a switch which executes the logic for server selection. It can divide the tasks into classes on basis of quality metric like burst time,etc. The Front end servers are designed to deliver the static pages mostly and in case of any query from the client the appropriate database server is connected. The routing and firewall switch ensures authorization and authentication and forbids any unintended user from accessing the files on the servers.

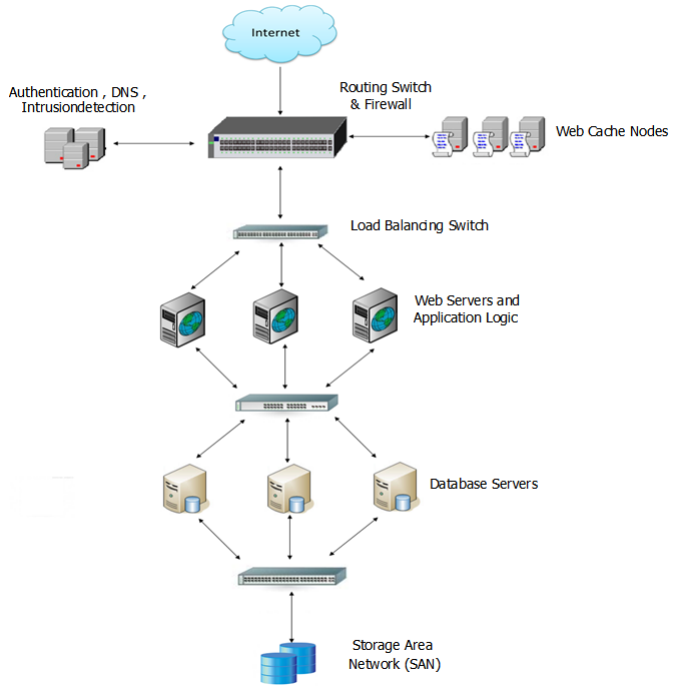

Figure 2: Depicting the 3 tier architecture for web servers.

A VOD system can be considered as a specific web server model where the database servers are replaced by the VOD servers. The Web servers serve the same purpose of delivering static content and some video content if present in the local cache. The system consists of 3 components [9,11]: the "set-top box" at the client's site, the distribution network, and the server. As with other networked systems, VOD can be designed as

centralized multimedia systems or distributed multimedia systems. A centralized VOD system places processing servers and media archives in a single site as a central node. Requests from clients are processed at the central node, and videos demanded are delivered through the network to the client sites. Centralized VOD systems are simple to manage, but they usually suffer from poor scalability, long network delay, and low throughput. The performance of centralized VOD systems can be improved if local servers are added. These local servers are also assumed to contain the most popular media.

The system architecture of a Video On Demand system basically consists of three major parts[11]: a client, a network, and a server. Each part can be subdivided further into components and interfaces. Figure 3 depicts the communication between clients and servers.

Figure 3: Communications Between Clients and Servers.

VOD system from the clients point of view is a simple operation. The user makes a selection from a list of available videos and the video is delivered to the user within the accepted Qos limits. Most networks use proxy servers or replicas to minimize delay .This is done by a process called request routing which directs the request to a particular web server on the basis of certain metrics. According to [7,8],there are 4 kinds of architecture for VOD networks[refer fig 4] a)Centralized, all the requests from the clients are handled at the original server, b)proxy based servers that are located close to the user end to reduce the load on the original server by caching, c) Content delivery networks, the servers are deployed close to the edge of the network to serve a fraction of clients request and d) hybrid, is basically a peer to peer approach.

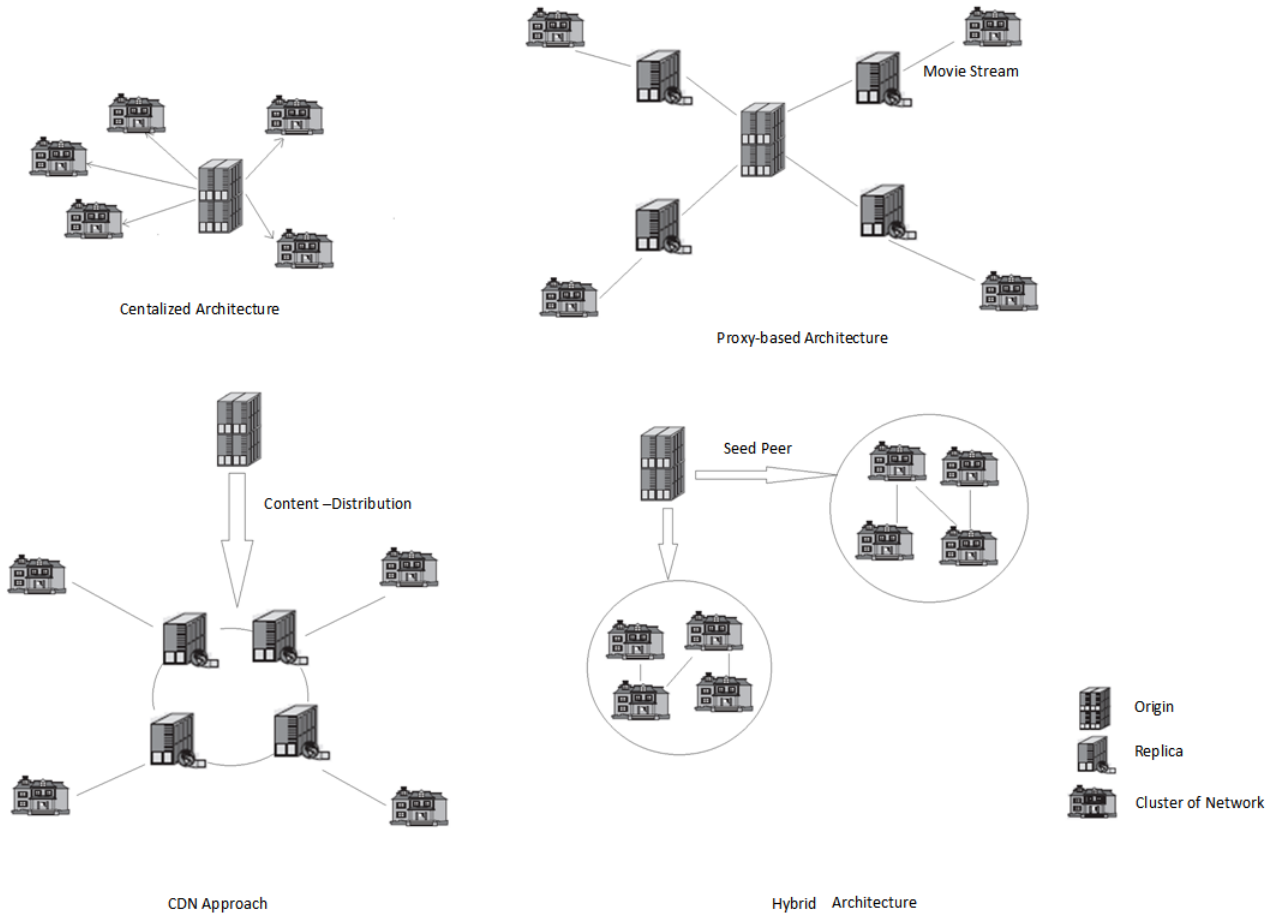

Figure 4: Depicting 4 kinds of architecture for VOD networks.

In our proposed solution we use a distributed architecture where the request first comes to a front end server from the client and after successfully passing through the authentication phase the video list is displayed in the web page and if the requested video is found in the page then the video is delivered to the client through local caching else it goes to a VOD server decided by the server selection strategy which then sends the desired video to the client. The arrival rate follows a Poisson distribution because that is the common mode of distribution for most of the internet traffic. The tasks are also assumed to have zero inter dependency among them. We neglect the different cost parameters and our sole focus is based on server selection keeping other parameters fixed. The detail sequence diagram for the centralized system is shown in the figure 5 The figure also shows different VCR functions like play pause,etc .

File Access models

This is useful during the video caching where the cache content is determined by the popularity or the hit ratio of the multimedia file. As time progresses, the cache content needs to be updated so that the cache contains the most popular video files. Previous studies have followed the Zipfs Law to calculate the popularity of the video files[12-14]. In Zipf-like distributions, the access frequency for a file of popularity rank i is equal to C/i^a, where C is a normalization constant and a(a>0) is the distribution parameter[8].The file usage patterns like which category of videos are accessed at which point of time during a day can also be analyzed and the cache be maintained accordingly.

2.3 VOD metrics

Many different metrics are used to evaluate the performance of a VOD server. They can be classified as Technology based and user based[9]. We have used Makespan and Resource utilization as two metrics for comparing algorithms. Makespan indirectly refers to the Response time of the system as a certain response time of say 0.5 sec implies that the requests should complete execution within 0.5 sec which suggests the makespan should not exceed 0.5 sec.

Resource utilization suggests what fraction of the total time a server is working. For a

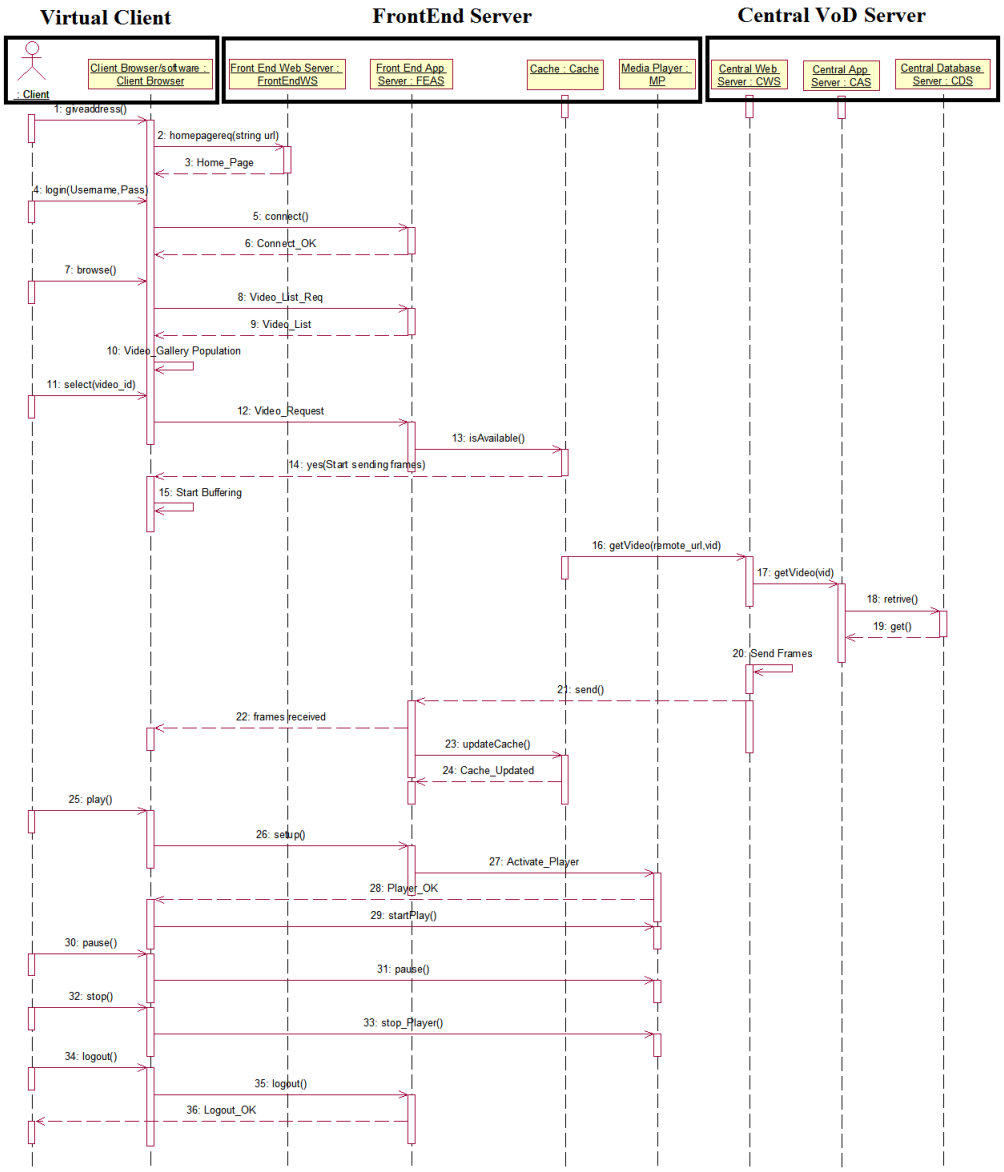

Figure 5: Depicting sequence diagram for a VOD connection and retrieval

single server

Ru=(Amount of time a server is idle)/Total time

While for a server system, the average resource utilization is the average of the individual resource utilizations.

Category	Parameter	Description / Example
Timeliness	Delay	Time taken for a message to be transmitted
	Response time	Round trip time from request transmission to reply receipt
	Jitter	Variation in delay or response time
Bandwidth	System level data rate	Bandwidth required or available, in bits or bytes per second. Basic mathematical models for
	Application level data rate	Bandwidth required or available, in application specific units per second, e.g. video frame rate
	Transaction rate	Operations requested or capable of being processed per second
Reliability	Mean Time To Failure (MTTF)	Normal operation time between failures.
	Mean Time To Repair (MTTR)	Down time from failure to restarting normal operation
	Mean Time Between Failures	MTBF = MTTF + MTTR
	Percentage of time available	MTTF / MTTF + MTTR
	Loss or corruption rate	Proportion of total data which does not arrive as sent, e.g. network error rate

Figure 6: Technology based Qos characteristics

2.3.1 Makespan

Makespan is defined as the largest completion time of all the tasks in the system. In thw VOD scenario, it is an indicator of the response time.For example, if the makespan of a

Category	Parameter	Description / Example
Criticality	Importance rating	Arbitrary scale of importance, may be applied to users, different flows in a multimedia stream, etc.
Perceived QoS	Picture detail	Pixel resolution
	Picture colour accuracy	Maps to colour information per pixel
	Video rate	Maps to frame rate
	Video smoothness	Maps to frame rate jitter
	Audio quality	Audio sampling rate and number of bits
	Video/audio synchronisation	Video and audio stream synchronization, e.g. for lip-sync.
Cost	Per-use cost	Cost to establish a connection, or gain access to a resource
	Per-unit cost	Cost per unit time or per unit of data, e.g. connection time charges and per query charges.
Security	Confidentiality	Prevent access to information, usually by encryption but also requires access control mechanisms
	Integrity	Proof that data sent was not modified in transit, usually by means of an encrypted digest.
	Non-repudiation of sending	Signatures to prove who sent or received data and when this occurred
	Authentication	Proof of identity of user or service provider to prevent masquerading., using public or secret encryption keys.

Figure 7: user based Qos characteristics

group of tasks exceeds a certain threshold then the tasks are not allowed as the response time Qos is not met.

2.3.2 Average Resource Utilization

Average Resource utilization for a system is defined as the average of the resource utilization r of various servers.
For a single server, utilization is given by
Resource utilization (Ru) = (Amount of time a server is idle)/Total time (3)

2.4 Existing algorithms

As described above the existing algorithms can be further divided into Traditional and Heuristic based algorithms. Traditional ones include First come first serve ,Random and genetic algorithms. There are another class of algorithms called the heuristic algorithms which comprises of min-min, max-min and weighted mean time scheduling [18]. These heuristics are applicable for heterogeneous task systems where we have servers of different capacity. This appears in an Expected Time Completion(ETC) matrix.The traditional heuristics have been compared with weighted mean time scheduling heuristic as suggested in [weighted] and the results are noted.

2.4.1 First Come First Serve:

This is a simple scheduling policy used in various load balancing servers. Whichever request comes first is served first irrespective of any other criteria. This algorithm though simple to implement has serious limitations. For e.g if a process with a high burst time is followed by a sequence of processes with low burst time then the latter has to wait for a long period of time to complete its execution. The response time for these processes is far greater than their burst times. So these processes undergo starvation.

2.4.2 Genetic Approach

The genetic approach doesn't refer to any predefined algorithm rather it refers to an algorithm framework. Different algorithm can take different amount of time for finding a solution to the problem. But all these approaches essentially comprises of 5 Steps- initialization,evaluation of firness function, selection, crossover and mutation. The details of the genetic algorithm are presented in the next chapter

2.4.3 Random

This is another scheduling policy where the tasks are distributed randomly to the available processors. If the distribution is truly random, then the random outweighs other algorithms in the long run. The figure below shows a comparative analysis of the 3 algorithms for load balancing.

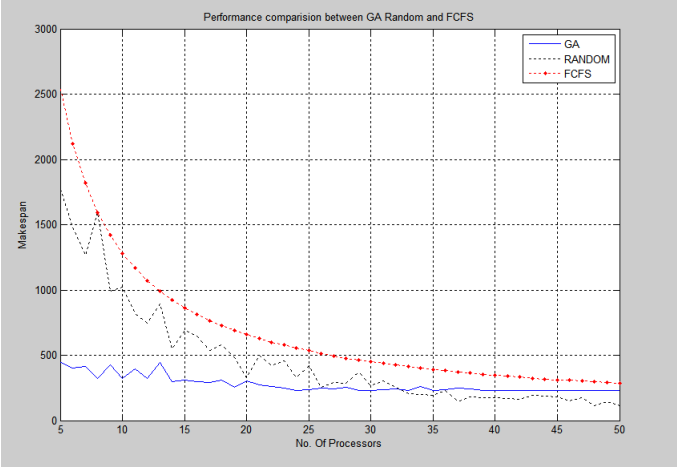

Figure 8: Comparative study of FCFS, GA and Random algorithms

Inferences

- FCFS graph is always higher than Random or GA which implies FCFS is the worst algorithm in terms of makespan.

- For a large number or servers, Random algorithms provide the best result, i.e the least makespan.

2.4.4 Heuristic methods

This consists of two phases [18,19]. First we choose a fixed arbitrary order and then for each task we choose the server with the minimum burst time.
In the second phase, the task with the minimum burst time among the group chosen is phase 1 is selected and assigned the corresponding server and the ETC matrix is updated with new completion times for the remaining tasks while the chosen task is deleted from the matrix. Completion time is given by the equation

CT(i,j)= ET(i,j)+ r(j)

Where rj is the ready time of machine j, i.e the time taken by the machine to complete all its pending tasks from the moment the task i is assigned to machine j. The maximum time to complete all the tasks is represented by the makespan.

Max-Min

This algorithm is similar to min-min except in the second phase the task with the maximum completion time is mapped first. This algorithm is known to provide better resource utilization than the Min-min algorithm.

Weighted mean time scheduling

The algorithm is adopted as described in [weighted] where the weighted sum of expected time is used. The weights are proportional to the servers capacity.

In our simulation setup, we used a task graph that consisted of 200X50 ETC matrix, i.e 200 tasks were assigned to be distributed to 50 heterogeneous servers. The result of comparison of the various algorithms for the given task graph are shown in fig 9 and 10

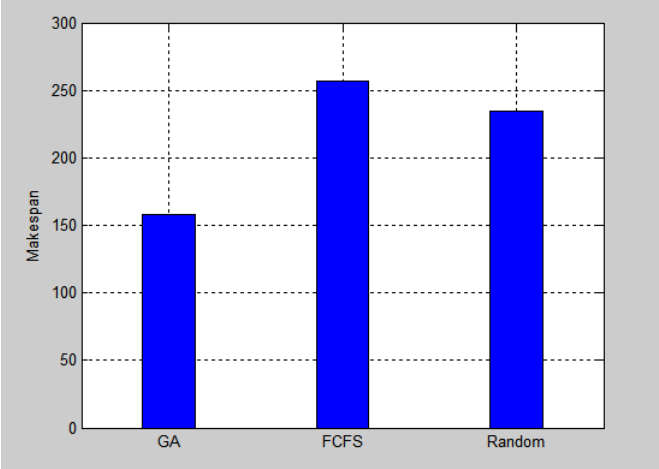

Figure 9: Comparative study of traditional algorithms on the basis of makespan

Inference

- The makespan is the highest for first come first serve and lowest for GA.

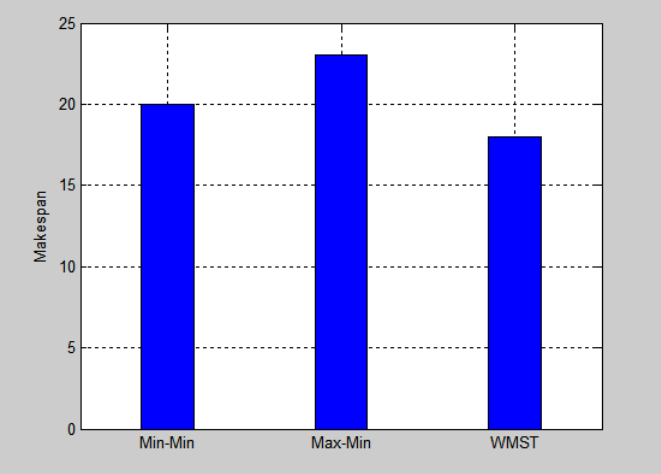

Figure 10: Comparative study of heuristics algorithms on the basis of makespan

Inference

- WMTS performs better in random taskset as compared to Max-Min and Min-Min.

2.5 Conclusion

A centralized architecture has a single video server, which becomes a bottleneck if the number of requests being processed by it increases. Such problem is handled efficiently by a distributed system. The widely used 3 tier architecture is widely used for web servers. For VOD servers other architectures like proxy Content Delivery Network(CDN) and Hybrid can be used. Different existing algorithms can be compared using average resource utilization and makespan as the metric on these architectures and the results are in coherence with the results derived from other work. But the effectiveness of these algorithms decreases as the size of task set increases. So a method to enhance the effectiveness of these algorithms has been proposed in the upcoming sections and the results obtained are compared with the existing algorithms.

Chapter 3

Heuristic Server Selection Strategies

3 Heuristic Server Selection Strategies

3.1 Introduction

Server selection is a problem of assigning a task Ti from a set of n tasks to the m available servers such that a performance metric is optimized. The need for server selection arises in case of a distributed system architecture. Choosing a good strategy is important because of the following reasons

1) It can reduce the overall cost of maintenance of the system.

2) It can reduce the response time of the system, thereby increasing customer satisfaction.

3) It can efficiently distribute the load among various servers and thus reduces the chance of breakdown of a particular system due to server overloading.

4) It can provide robustness and easy scalability to the system.

So selecting a good strategy is of paramount importance. But each of the existing algorithms do not apply well to all the scenarios as discussed in the previous chapter.

3.2 Genetic algorithm

Some of the existing algorithms are

1. First come first serve
2. Genetic
3. Random

The genetic algorithm is an optimization technique that has it base on the basis of natural selection. A GA consits of candidates or population which evolve based on some predefined rules such that each evolution produces a better population(i.e population which minimizes the cost function).

Some of the advantages of GA are

- It optimizes both continuous and discrete variables.
- It simultaneously searches from a wide sampling space.
- It is well suited for parallel computing.
- It optimizes complex cost functions quite well(there are several local minimas) and produces the global minima.
- It provides a list of optimal solutions not the single best optimum solution.
- Encoding the variables are easy when they are represented in terms of genes.

GA works in the following steps

- **Initialization:** An initial population of random fitness values is generated here.
- **Evaluation of fitness function:** This function decides which are the genes to be propagated to the next generations.
- **Selection:** The unfit strings are removed and the other strings are selected for the following phases of GA.
- **Crossover:** This operation is done in order to find a local minimum.
- **Mutation:** This operation is done in order to switch to another local minimum in the search space. This is the power of GA that guarantees a global minimum value among the search space.

A. Initialization

First, in this step we represent the problem in GA format. It consists of two phases[1]

- **String representation**
- **Initial Population**

String representation

All the search nodes are in the search space are represented by unique strings, so choice of strings for GA representation is important. There are many popular presentations like binary, natural number representation etc. We have taken natural number representation of strings in this thesis.

In this thesis we are selecting tasks from a task graph. As the search space is large we apply the GA to a smaller number of tasks and find the near optimal allocation. For this we have taken a sliding window technique. The window size is fixed and tasks are taken from the main task graph to this window and accordingly GA is applied to this window. We have fixed the task graph and taken only the indices of tasks to the window not the actual burst time values (Refer to Fig 11)

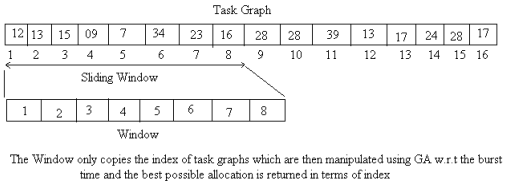

Figure 11: Depicting the task graph along with the window size to be used.

So the string representation becomes easy, for e.g. the initial contents of the window becomes the initial gene to be operated and all the operations are performed with respect to the actual burst time of the tasks not the indices. A separate Index is maintained in order to locate which task goes to which processor (Refer Fig 12) .

It clearly shows which task is mapped to which processor e.g. task 6 is mapped to Processor 2

Initial population

All the tasks, indexes of which are present inside the window become the initial population and population size is same as window size. Then each of the tasks are taken from window and randomly allotted to any processor, corresponding mapping is stored in the string and the index. Here we have taken no. of strings inside the population same as the no.

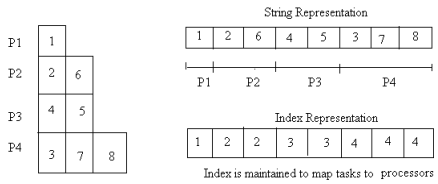

Figure 12: Depicting the allocation of tasks to the available processors.

of processors. E.g. in the above example our initial population will contain 4 strings each of which will be showing a different mapping to the processors.

B. Evaluation of Fitness Function

The objective function or fitness function is the most important aspect of any of the optimization problem. Depending up on the fitness function the solutions are accepted or rejected. Designing the fitness function for the GA allocation is more important as it reduces the search space drastically. So in a less number of iterations we can reach nearly optimal solution. Parameters affecting fitness of GA are

- **Makespan**
- **CPU Utilization**
- **No. of processors**

Makespan

The first objective of the GA algorithm is to minimize the makespan. Makespan is the largest completion time of all the tasks in the system. Based on the above example makespan is calculated as follows (Refer Fig. 13).

Burst times are taken from the actual task graph and indices are taken from the strings of populations. This is the makespan for one of the four strings present in the population. In this thesis we are allocating the new tasks only after all the tasks represented in the window are executed successfully , hence we assume zero initial loads on the processors and makespan is calculated based on the current active load only.

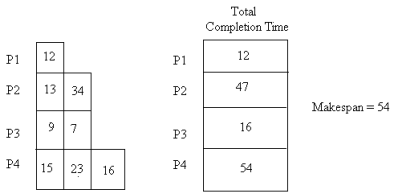

Figure 13: Calculating the make span.

CPU Utilization

This is another useful parameter as it decides whether the load is properly balanced across all the processors or not. High average processor utilization implies that load is balanced properly. Individual utilization of processors are calculated by the formula given by Eq.1.

Utilization (P) = Completion (P) / Makespan (1)

Where P denotes the processor for which utilization is being calculated. So the utilization for the previous example is calculated as follows.

Utilization (P1) = 12/54 = 0.2222

Utilization (P2) = 47/54 = 0.8704

Utilization (P3) = 16/54 = 0.2963

Utilization (P4) = 54/54 = 1.0000

Then dividing all the individual utilization by the sum of the utilization gives the average CPU utilization.

Average CPU Utilization = (0.2222+0.8704+0.2963+1.0000)/4 = 0.6

Number of Processors

This is another deciding factor for the fitness. We can say that if number of processors will be high then the fitness will be reduced as we are calculating fitness per processor. Hence the combined fitness function is given by Eq. 2.

Fitness= (1/makespan)x (average utilization)x(1/no.of processors) (2)

C. Selection

The selection procedure followed in this thesis is purely based on the probability of the fitness values of the strings. After calculating the fitness values of each strings probability of each string is calculated by the ratio of fitness value to the sum of fitness value. As better strings have better fitness values the chances of these strings for the survival to the next generations will be more hence the strings having greater fitness values will represent high probability. We have set the limit for this probability. i.e. the strings having probability value less than 0.2 will be rejected for the next generations. The fitness and the probability values are thus calculated (Refer Figure 14).

As per our strategy string1 and string 4 are rejected for the next generations.

String	Fitness Value	Probability
1	0.0028	0.0023
2	0.5874	0.4867
3	0.3822	0.3167
4	0.2345	0.1943
Total	1.2069	

Figure 14: depicting the fitness value and probability of various strings

D. Crossover

The crossover technique that we have followed in this thesis is PMX Crossover. PMX Crossover is a genetic algorithm operator. For some problems it offers better performance than most other crossover techniques. Basically, parent1 donates a random swath genetic material and corresponding swath from the other parent is sprinkled about in the child. Once that is done, the remaining alleles are copied directly to the child. Randomly select a swath of alleles from parent1 and copy them directly to child. Note the indexes of segment. Looking in the same segment positions in parent2, select each value that hasnt already been copied to the child .For each of these values:

Note the index of this value in Parent 2. Locate the value V, from Parent 1 in the same position.

Locate this same value in parent2.

If the index of this value in Parent 2 is part of the original swath, go to step1 using this value.

If the position isnt part of the original swath insert that value into the child at this position.

Copy any remaining positions from parent2 to child.

PMX Example:

Parent1: 8 4 7 6 5 1 9 10 2 3

Parent2: 3 10 1 6 4 9 5 7 8 2

Child 1: - - - 6 5 1 9 10 - -

- We copy a random swath containing consecutive alleles from Parent1 to Child.

- 4 is the first value in the swath of parent 2 that isnt in the child. We identify 5 as the value in the same position in parent 1.We locate the value 5 in parent 2 and notice that it is still in the swath. So we go back to step one with 5 as the value.

- Repeating step 1 once again we see that 9 is in the same position in parent 1 and we locate 9 in parent 2. it is also in the swath, so we repeat step 1 once more with 9 as the value.

- Repeating step 1 we see that 1 is in the same position in parent 1 and we locate 1 in parent 2 in the 3rd position .Finally we have obtained a position in the child for value 4 from step.

  Parent1: 8 4 7 6 5 1 9 10 2 3

  Parent2: 3 10 1 6 4 9 5 7 8 2

  Child1: - - 4 6 5 1 9 10 - -

- 7 is the next value in the swath in parent 2 that isnt already include in child. So we check the same index in parent 1 and see 10 in that position. Now we check for 10 in parent 2 and find it in 2nd position. Since second position is not part of swath, we have found a home for value 7.

  Parent1: 8 4 7 6 5 1 9 10 2 3

  Parent2: 3 10 1 6 4 9 5 7 8 2

  Child1: - 7 4 6 5 1 9 10 - -

- Now we have considered all the swath values, so everything else from parent2 drops down to the child.

  Parent1: 8 4 7 6 5 1 9 10 2 3

  Parent2: 3 10 1 6 4 9 5 7 8 2

  Child1: 3 7 4 6 5 1 9 10 8 2

The same procedure can be repeated again to obtain the second child only the parents are swapped. The crossover is done on the existing strings of the population to prepare the next generation strings. Crossover operations guarantee to give a local minimum value from the search space.

E. Mutation

This is the operation that guarantees a global minimum from the set of the local minimas. This operator brings diversity in the search space so that solutions can be picked from different regions of the solution space. Generally mutation will be done less as it fluctuates through out the search space. If we allow 10 generations then crossover will be done all times but mutation is done only 2 times which represents the less operations of mutation in GA. In this thesis we have followed swap mutation procedure in which we take randomly two strings which are different from each other and swap the values at a randomly generated position.

Instead of waiting GA to converge we allow it to run for 10 generations and after that the fittest string is taken for task allocation.

Window updation

After generation of the fittest allocation the window is shifted to next set of task indexes from the task graph and the same procedure is repeated once again. As we are following a deterministic approach i.e. length of task graph is known prior to the invoking of GA there may be a case when tasks will fall short of window size. At that scenario we simply allocate them using FCFS policy.

3.3 Max-Min

There are another class of algorithms called the heuristic algorithms which comprises of min-min, max-min and weighted mean time scheduling [18]. These heuristics are applicable for heterogeneous task systems where we have servers of different capacity. This appears in an Expected Time Completion(ETC) matrix.

Max-min

This consists of two phases [18,19]. First we choose a fixed arbitrary order and then for each task we choose the server with the minimum burst time.

In the second phase, the task with the maximum burst time among the group chosen is phase 1 is selected and assigned the corresponding serverand the ETC matrix is updated with new completion times for the remaining tasks while the chosen task is deleted from the matrix. Completion time is given by the equation

CT(i,j)= ET(i,j)+ r(j)

Where rj is the ready time of machine j, i.e the time taken by the machine to complete all its pending tasks from the moment the task i is assigned to machine j. The maximum time to complete all the tasks is represented by the makespan.

3.4 The proposed Solution- Ga-Max-min Algorithm

This algorithm merges the genetic algorithm and the max-min algorithm. This results in the enhancement of the performance of the genetic algorithm. So this kind of algorithm can be used where the number of tasks is very large.

Ga-Max-min algorithm

1) For all tasks i in the task set
2) Divide the tasks into classes on basis of burst time or previous history
3) Send the tasks to the appropriate queue
4) Apply different selection algorithms as applicable to the different queues
5) Makespan=Calculatemakespan(Task set)
6) Resource utilization =Calculate resource utilization (Task set)
7) End

The task set is divided into classes based on burst time or previous history. Then for each queue Resource utilization and makespan is calculated by the calculate resource utilization() an d calculate makespan() functions. Both these functions take task set as the input.

In our simulation setup, we used a task graph that consisted of 200X50 ETC matrix, i.e 200 tasks were assigned to be distributed to 50 heterogeneous servers.

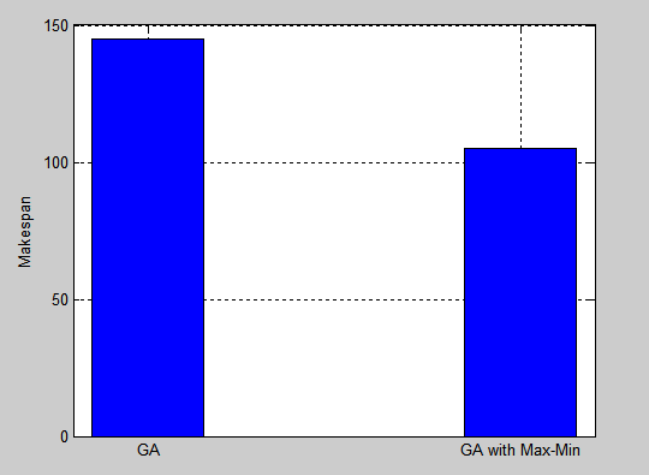

Figure 15: Comparative study of GA and GA-max-min algorithm on the basis of makespan

Inference

- The composite algorithm GA-max-min performs better than GA.

3.5 Average resource utilization

Average Resource utilization for a system is defined as the average of the resource utilization of various servers.

Resource utilization (Ru) = (Amount of time a server is idle)/Total time

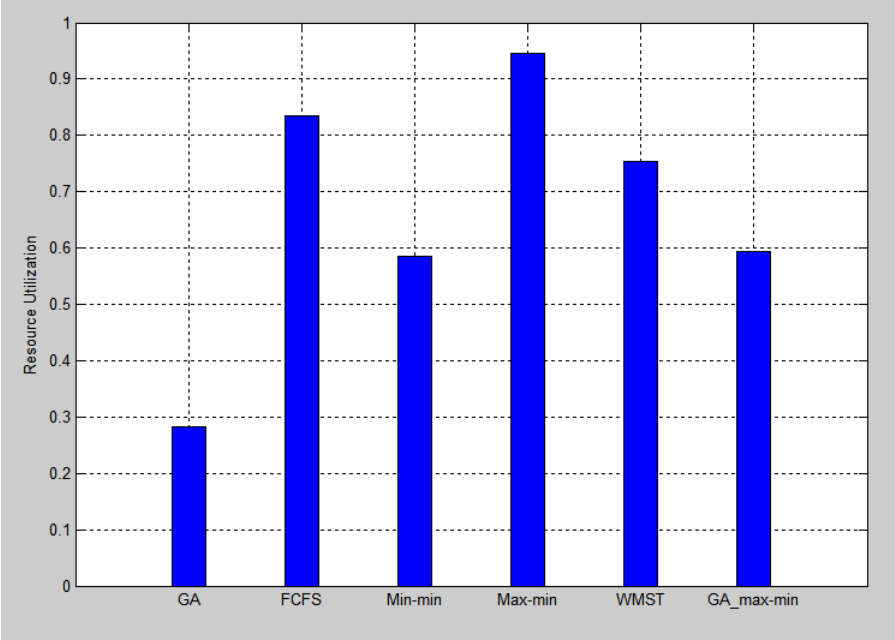

Figure 16: Comparative study of various algorithms on the basis of Average Resource utilization .

Inferences

- The resource utilization is highest for Max-min and lowest for Genetic algorithm which shows that max-min uses the resources in a more efficient way.
- The composite algorithm GA-max-min helps enhance the performance of GA by improving the resource utilization from 0.28 to 0.59.

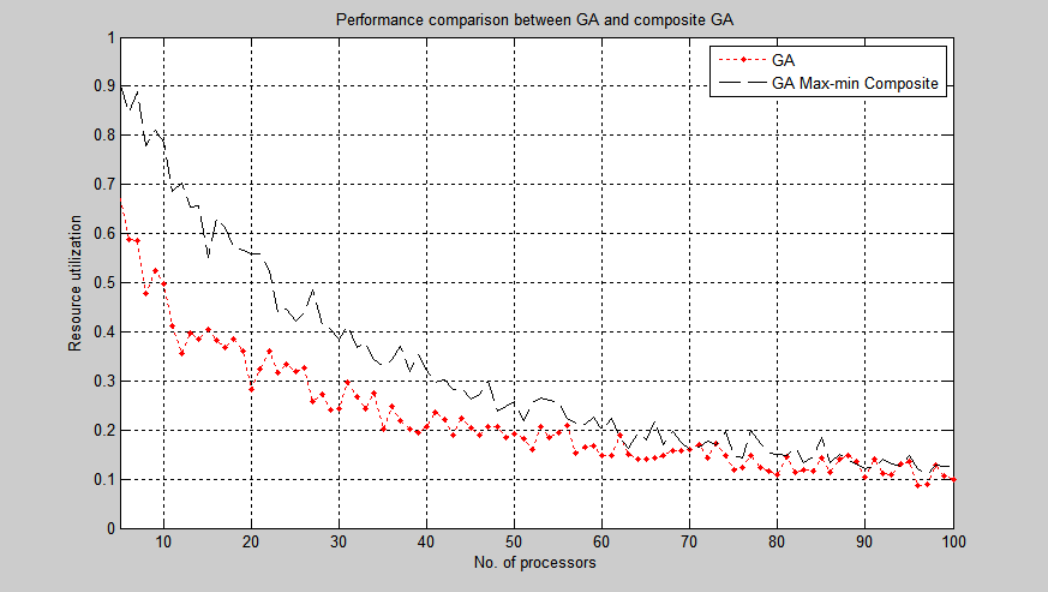

Figure 17: Comparative study of Ga and Ga max min over resource utilization and no of processors

Inferences GA-Max-min is more efficient at utilizing resources than GA for upto 50-70 number of processors after which the two algorithms produce similar results.

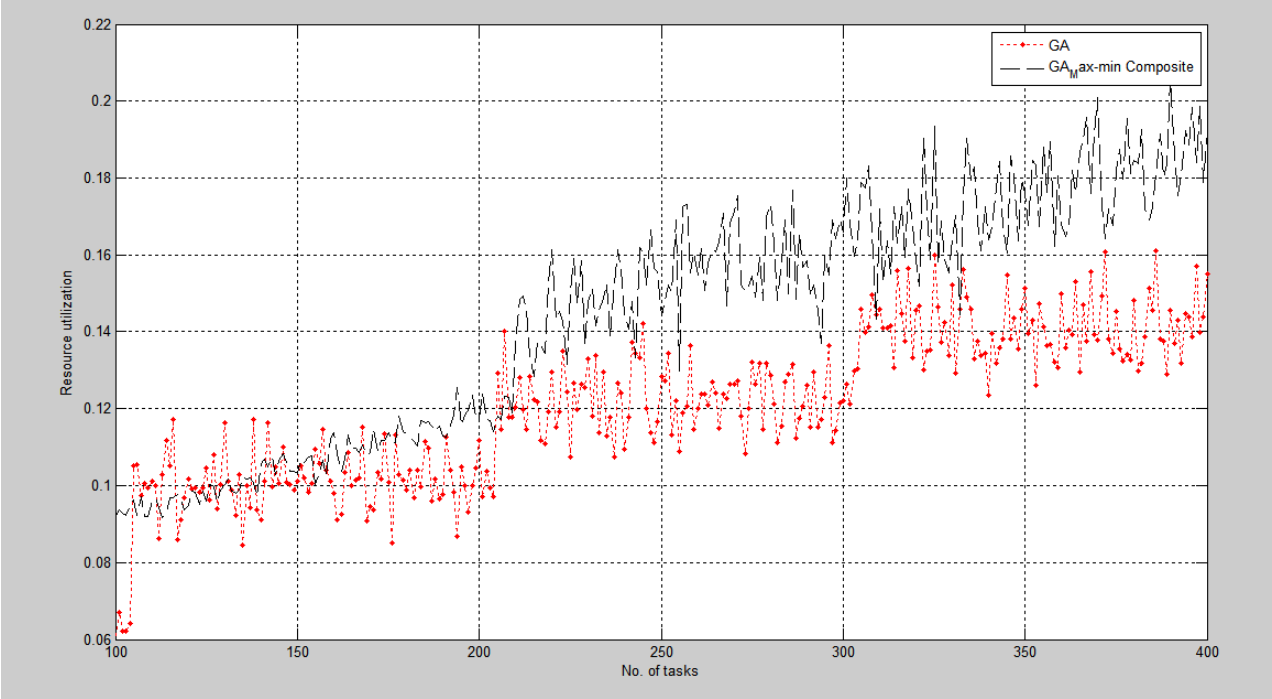

Figure 18: Comparing GA and Ga-max-min over resource utilization when no of tasks varies.

Inferences

As the number of tasks increases, the composite algorithm provides better resource utilization than the original GA algorithm. So the proposed solution will generally yield better result for large amount of tasks where other algorithms do not give good results. For tasks greater than 200 and processors greater than 50,the composite algorithm outperforms Genetic algorithm.

3.6 Conclusion

This chapter essentially describes the two best algorithms for large set of tasks-GA and max-min. GA was found to be providing the lowest makespan which was due to the learning and rejection of the unfit candidates while the algorithm progressed. Max-min was found to be efficient in utilizing the resources well. So a combination of GA and max-min known as Ga-max-min was proposed aimed at taking advantage of both the algorithms. The proposed solution worked well for large number of tasks. In fact the graph shows that the when the number of task is low, then there is not much difference between Ga and the proposed Ga-max-min. But significant increase occurs in resource utilization as the number of task increases. Such algorithms can be combined to produce an algorithm framework which can deploy various selection strategies dynamically as the request arrival rate varies.

Chapter 4

Conclusion

4 Conclusion

4.1 Conclusion

In this thesis we compared the various server selection algorithms like FCFS, Random, Min-min on the basis of makespan and Average resource utilization. We also combined two heuristics Genetic algorithm and max min to get a new heuristic GA-max-min. We chose Genetic algorithm as one component of the combined heuristic as it was feasible to apply genetic algorithms for large data sets and the max min algorithm as another component as it provides the best resource utilization. The new heuristic had intermediate values for both resource utilization and makespan. For unpredictable nature of the tasks genetic algorithms works best as the system learns about the nature through GA and thus utilization is sometimes low. The combined heuristic can, thus be used to enhance the Average resource utilization of the Genetic algorithm and it also decreases the makespan that the genetic algorithm produces.

Chapter 5

Future Work

5 Future work

5.1 Future Work

The thesis analyzed the performance of various server selection algorithms based on two performance metric, Average resource utilization and makespan. In future other parameters like scalability, throughput can be used to analyze the performance of these algorithms. Different weights can be assigned to different metrics. Based on the scenario and the nature of tasks, the weights to the different metrics can be adjusted. For example in a scenario where response time is the most important criteria like the video on demand system, makespan can be given the maximum weight. Basing on the weights different algorithms can work well for different circumstances. The algorithm which has the best weighted value can be used for t server selection Further a complete algorithm framework can be formed by combining various algorithms as an extension of Ga-max-min which caters to varying nature of the tasks and the switch can dynamically change algorithms as the request rate varies. Real time data can be collected on the performance of the algorithm framework with real data and can be compared with the simulation results. The algorithm framework can be fine-tuned from these real life observations through some soft computing measures like Learning Automata and Neural networks.

References

[1] A. Y. Zomaya, and Y. H. Teh. *on using Genetic algorithms for dynamic load balancing.* IEEE transactions on Parallel and Distributed Systems, vol. 12, no. 9, September 2001. (references)

[2] Z. Zhang and W.Fan. *Web server load balancing: A queueing analysis.*European Journal of Operational Research, vol. 186, no. 2, pp. 681-693, April 2008.

[3] V. Gupta, M.H. Balter, K. Sigman and W.Whitt. *Analysis of join-the-shortest-queue routing for web server farms.*Performance Evaluation,vol. 64, no. 9-12, pp. 1062-1081, October 2007.

[4] D. Niyato and C.Srinilta. *Load balancing algorithms for Internet video and audio server.* Proceedings of 9th IEEE International Conference on Networks, pp. 76,2001.

[5] J.L. Wang, L.T.Lee and Y.J.Hunag. *Load balancing policies in heterogeneous distributed systems.* Proceedings of 26th Southeastern Symposium on System Theory, pp. 473-477, 1994..

[6] G.Ciardo, A.Riska and E.Smirni. *EQUILOAD: A load balancing policy for clustered web servers.* Performance Evaluation,vol. 46, no.2-3, pp. 101-124, October 2001.

[7] F.T houin. *VOD equipment allocation.*Tech. report,Mcgill University,Montreal, Canada.

[8] F.Thouin and M.Coates. *VOD networks:Design approaches and future challenges,Proceedings of Network IEEE.*pp. 42-48,montreal,March-april 2007.

[9] N. Panigrahi and B. Sahoo. *Qos based retrieval strategy for video on demand.*Available online at http://dspace.nitrkl.ac.in:8080/dspace/bitstream/2080/789/1/bdsahoo-2009.pdf. Last visited may 08 2011.

[10] N. Carlsson and D.L.Eager. *Server Selection in large scale Video on Demand.*Proceedings of ACM transactions on multimedia,computing and communications,February,2010.

[11] M. Ko and I. Koo. *An Overview of Interactive Video On Demand System.*Technical Report, The University of British Columbia, Dec 13,1996.

[12] N. Jian et al. *Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network* .Proc. IEEE Intl. Conf. Commun. (ICC), Anchorage, AK, May 2003

[13] B. Wang et al. *Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution* .Proc. IEEE Infocom, New York, NY, June 2002.

[14] T. Wauters et al. *Optical Network Design for Video on Demand Services.* Proc. Conf. Optical Network Design and Modeling, Milan, Italy, Feb. 2005.

[15] D. Ligang, V. Bharadwaj, and C. C. Ko. *Efficient movie retrieval strategies for movie-on- demand multimedia services on distributed networks.*Multimedia Tools Appl., vol. 20, no. 2, pp. 99133, June 2003.

[16] M. Guo et al. *Selecting among replicated batching video on demand servers.*Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video,May 2002.

[17] S. Sharifian, S.A. Motamedi and M.K. Akbari. *A predictive and probabilistic load balancing algorithm for cluster based web servers.*Proceedings of Applied soft computing, pp. 970, Jan 2011.

[18] S. S Chauhan and R.C.Joshi. *A weighted time min min max-min selective scheduling strategy for independent tasks on grid.* Proceedings of Advance Computing Conference(IACC),Patiala,India, Feb 2010.

[19] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. Freund. *Dynamic mapping of a class of independent tasks onto heterogeneous computing systems.* 8th IEEE Heterogeneous Computing Workshop (HCW '99), pp. 30-44, San Juan, Puerto Rico, April 1999.

[20] A. Narasimhan, *Distributed multimedia applications-opportunities, issues, risk and challenges: a closer look* .IASTED International Conference on Intelligent Information Systems , pp.455-460, 1997

[21] http://www.findmyhosting.com/bandwidth-explained/,Last visited 16th march 2011.